Agile software development and user centered design processes: can they co-exist?

Background

There is an imaginary line in interactive systems with the user on one side and the software on the other. This imaginary line is the physical interface between these two elements – often called the user interface but more correctly called the user-computer or human-computer interface (the interface between these two subsystems). This imaginary line also separates the people who work in the field into those focused on the user side of the line (the fields of Human Computer Interaction, Human Factors Engineering, Visual Design, Interaction Design, and Industrial Design) and those that focus on the hardware and software side of the line (Software Engineering, Hardware Engineering, Electrical Engineering, and Mechanical Engineering). Though some people try to work on both sides of the line, the separation is real–the professional backgrounds of are different, the techniques used are different, and the focus of the activities are different.

In attempting to document an effective process for the design and development of interactive systems, some processes attempt to ignore this line or believe that one side can address the issues of the other. This is a mistake, and there is no better example of this issue then in the on-going discussions of the relationship between the agile software development process and the user-centered design process.

There are a number of processes that have emerged from software engineering. The linear based “waterfall” model was the first recognized process. It’s origin likely dates back to as early as 1956, but is generally believed to have been formally defined in a 1970 article by Winston Royce. Recognizing the limitations of this model, particularly the concepts that any step in a process must or even could be completed before moving forward and without the need to go back to earlier steps in the process, led to the development of other models such as the modified waterfall model and the spiral model (Boehm, 1988). However, these models did not adequately address involvement of users in the processes to ensure that users needs are met. This led to a second wave of processes developed in the 1990s.

In 1995, Milington and Stapelton proposed the rapid application development approach as an alternative to the waterfall and spiral models that attempted to

incorporate a more user centered view. Time cycles for projects were modified in a concept called “time–boxing”. The idea was to develop a series of small projects within a larger project. User-focus was provided through workshops (known as Joint Application Development workshops) in which users and developers worked together to come up with requirements. Between the two changes introduced by this new process, separation of a large project into smaller deliverables has had greater benefit to the overall development process than the joint application development workshops.

Capitalizing on the concept of dividing projects up into smaller local projects, a series of new processes were developed within software engineering that included DSDM, eXtreme, Scrum, Adaptive Software Development (ASD). These new processes were known as “agile software development processes.” The intent of these “agile” processes (and the source of this name) is to relieve the software development of the many burdens of a rigid process that hampers the ability to produce software on time and within budget. There were 12 main principles that define the original “agile manifesto” published in 2001. The principal characteristics within the manifesto can be grouped into four main areas: (1) better team communication (small teams, co-location of teams, daily communication), (2) iterative delivery (“working software” as a measure of progress, rapid and continuous delivery of “useful” software, deadlines measured in weeks instead of months or years), (3) flexibility (self organizing teams to meet the needs of a specific project, the willingness to accept and address late and changing requirements), and (4) accountability (process transparency, integration of members outside the main development team – though originally only “business people”). But Agile is on the computer side of the imaginary line. However, though there were attempts to incorporate users into these processes, all of the processes described above came from software engineering and have software development issues as the core of their perspectives. In other words, though they acknowledged the need to address users and user requirements, they emanated from the computer side of the human computer interface and naturally focus on issues associated with implementation.

Developed from a perspective on the human side of the line were the “user centered design” processes (also known as the human centered design process). These processes emerged from the fields of Human Computer Interaction and Human Factors Engineering and, though the included the implementation of a design, they focus primarily on the developing the specification for the user interaction. Formal definitions for this process were laid out in MIL-STD46855: Human Engineering Requirements for Military Systems, Equipment, and Facilities (1994); in two international standards organization (ISO 13407: Human-centered design processes for interactive systems and ISO 9241:Ergonomics of Human System Interaction), (both 1999); and in some commercial practices (e.g., LUCID, the Logical User Centered Interaction Design process described by

Cognetics Corporation). These processes share some of the same characteristics as the software processes described above. Rapid development and delivery of designs with an opportunity for feedback is central to these approaches as well. Multidisciplinary teams are used (though different disciplines than are involved in the software engineering developed processes). And transparency is provided between the team developing the interface design and those who would be responsible for implementing the design. However, there are some key differences.

Within the user centered design processes, there is both a divergent and a convergent phase. In the divergent phase, multiple designs are created and presented for evaluation. The user centered design approach relies on simulation or the creation of mockups over the development of working software. Included in the simulations or mockups are only those portions of the interface that are being evaluated and the simulations or mockups do not have to follow the requirements for robustness, extensibility, maintainability, coding practices, or code documentation–all of which are necessary for fielded code and considerations in the agile software development process. During the convergence phase, results from evaluations of alternate designs are used to create a single, unified design that is likely to be a combination of previously considered designs concepts.[1]

The Issue

In some circles, it is believed that the agile software development process incorporates both the concepts of good user interface design and good software development. It is assumed that obtaining feedback from end-users or end-user representatives during the implementation phase provides sufficient feedback to ensure an appropriate interface design is created. There are several issues with this assumption.

Consider the often-used analogy of building a house. If you wanted to have a house built that is custom designed just for you, no one would ever consider hiring electricians, plumbers, carpenters, etc. and asking them to begin building a house. "Just get started building and we'll figure things out as we go." If such a case were to occur, all members of the team might get together to discuss what they're planning on doing and then begin moving forward, but without sufficient detailed documentation of the intended endpoint, multiple conceptual models (e.g., designs) will exist within the minds of each person. In addition, these design are not likely to be completely thought-out or consistent (either between designs or even within a single design). The outcome is unlikely to produce an acceptable solution. Numerous past examples of this problem can be cited. The

---

[1] This does not occur in an agile software development process since the cost of developing multiple operational versions, even partial versions, is not considered.

author of this paper has routinely experienced cases where work progressed on individual elements of the design and then the new design elements had to be forced together to create some kind of a final solution. Or the design contained essentially separate elements of functions that only coexisted in the interface but did not truly integrate (the term “surface integration” was coined for this effect nearly 20 years ago). In any case, the solution was not optimized for the user experience. There is a classic cartoon from 1970 of developing the tire swing that demonstrates how this lack of a shared vision results in problems for all members of the team [2].

Now consider the effect if you add the concepts of agile to this process. Ensuring that the team is small and communicates regularly will certainly lessen some of the issues. But the Agile team is first and foremost a development team. Including an Interaction Designer or similarly skilled person on the team, even from the project outset, only provides another design concept to the development team to consider (albeit one that is likely to be more detailed and consistent in the user interaction). But it still remains within only the mind of the architect – the feedback opportunity within the agile process is after an incremental build. And seeing a partially complete version of the product does not provide a clear enough picture of where the product is ultimately heading in either overall concept or detail. It is possible to look at a partially complete design and tell whether or not it is heading in the direction intended only if you know where is supposed to be headed. As a result, viewing a partially complete product may result in each member of the team falsely believing they are all working on the same concept because they are evaluating it against their own vision of the final destination. This is compounded by the fact that final destination is still not shared from the project outset.

Assuming that someone is able to clearly identify an “issue” or desirable design change to improve the user interaction, there is the issue of having already committed to a partial solution. In construction, the rule is “measure twice, cut once.” It is not possible to un-cut wood or un-pour concrete. Similarly, the ultimate loss in time and cost in even an agile software development process prohibits making significant changes; so compromises to the user interface design will nearly always be made over changes to the existing, partially complete software structure. This fundamentally changes the goal of the software design from top-down design approach (the user interface design driving the development of a solution) to bottom-up design approach (the software solution driving the user interface design).

Also consider the risk of software development complexity. There is always the possibility that software development will encounter an issue in the implementation. To maintain the Agile schedule, some portions of the design

---

[2] See http://www.businessballs.com/treeswing.htm

may not be competed in one iteration and moved to another. This is not an issue for stakeholders to observe and approved progress, but it is a significant issue for end users who would now have to experience both a partial design and a partially functioning design.

The Solution

Clearly the solution is to have a detailed vision, shared across the entire team, prior to beginning the agile software development process. In other words, the focus on iterative deliverables development over documentation is fine *provided this lack of documentation does not include the user interface*. This is the reason that many teams, even those that follow an agile software development process also follow a user-centered interface design process for the presentation layer (the view of the software from the user's perspective).[3] In fact, nearly every agile development-based project this author has had experience with has included wireframes prior to any code development. This makes sense since it does not make sense to write code without a visual concept to work against, but these wireframes extend only to the specific iteration and do not represent a full design. The solution is to extend the wireframe development to the entire design to ensure a complete and consistent design prior to the first iteration of the Agile process.

Accepting the assumption that there needs to be a user centered design process and a separate agile software development process, the question of the relationship between these two processes remains to be defined. One possible interaction between these two processes is that the user centered design process produces an outcome that covers all possible scenarios and use cases, all of the logic defining the interaction, identifies all edge cases, etc. sufficient to constitute an interface specification. Note that the skill set associated with the user centered design process does not include software programming or web development (though the team needs to be knowledgeable enough to know what is feasible and transparency with the development team is intended to ensure this occurs). Also note that the user centered design process is intended to explore unknown and new areas of the interface design, areas known to need redesigning to address known interaction or experience design concerns, or areas of perceived interaction or experience design concern. It is not intended to address areas where these problems do not exist. For example, login processes and checkout processes may be assumed to be correctly designed and may not be part of a user centered design process. These factors suggest that it is ill advised to place the responsibility of full interface specification, even limited to the presentation layer, on the user centered design team.

An alternative approach would be for the user centered design process to

[3] [insert references and examples of processes from AOL, EightShapes, etc.]

produce guidelines to the agile software development process, but not produce an actual design. Having the agile software development process responsible for completely defining the design, even when provided with guidelines, would eliminate valuable data obtained from the user centered design process on elements of the design, its interaction, architecture, naming conventions, navigation model, etc. – the elements that would have been tested through a simulation or mockup. In addition, as mentioned previously, the agile software development process does not include a divergent process needed to explore design options before settling on a single interface design solution.

The third alternative is that the user centered design process produces a concept for select portions of the design–as mentioned previously that portions of the design would be new elements of the design, areas of known interaction or experience design issues, or areas of suspected interaction or experience design issues. What the user centered design team produces is a reasonably complete specification for a significant portion of the design. The full logic, edge cases, etc. that are necessary to complete the design are produced as part of the agile development process. The agile software development process with iterative implementation and feedback is suited for this responsibility if the feedback process includes representatives of the design team. It becomes apparent in comments on the agile development team to be aware of when assumptions are being made to complete the design concepts provided by the user centered design team and to obtain their feedback that these assumptions are consistent with the design concept. It is incumbent upon the user centered design team to provide rapid feedback to the agile software development process so as to have minimal impact on the software development progress. However, there is still a significant set of risks in this approach. The agile software development process is predicated on a small team of individuals. This small team would not be capable of addressing an enterprise-wide project in a timely fashion. Multiple agile software development teams would suffer the same problems of inconsistency between the teams that would have a detrimental effect on the user interface. Therefore, the successful application of this process of partial specification prior to development is likely to have very limited applications. Even if the project is small enough, there is still a reasonable probability that initial implementation may have to be radically changed to support other features if there are dependencies between these features – which is quite often the case. Unless sufficient analysis has been done on the overall concept prior to performing any development, the order of implementation in an incremental build may increase the probability of major change (or compromised in the design). In other words, the order in which incremental builds implement specific features may have a significant impact the probability of needing a major change in the software's architecture.

The issue with all of these potentials for interacting between a user centered

design process and agile software development process continues to show that the user interface is likely to be compromised as a result of attempts to them for granted without a complete and clear specification of the user interface and behavior. Since this is not the outcome of the user centered design process more possible through an agile development process, this suggests that there is an interim engagement between the two processes – a joint engagement to complete the work of both sides. If, at the end of a user centered design process, the interface or user experience design team in conjunction with the software development team then jointly completes the partially completed design, issues of both sides can be addressed. The compromises may need to be made to the user interface at this point to address implementation issues, but it is done with the full awareness of the team responsible for developing the design and at a point where there is minimal cost associated with these changes since modifying the concept or even a mockup or simulation is far easier than restructuring operational code. Therefore, this purpose of this interim step is the creation of a full specification of *just the user interface,* addressing the needs of the interface designers as well as the needs of the developers. And the outcome is a specification that is detailed enough to allow development to progress with minimal chance of having to make major changes or sacrificing the integrity of the design.

However, even in a well thought out process, there will exist some probability that there will be a need for additional user research or possible additional interaction or experience design work. In these cases, a disruption to the agile software development process may be inevitable, but it is minimized by two factors–the user centered design process itself is agile or like in its nature and the agile software development process may focus on other elements of the design answers are obtained. In some cases, the answer to these questions may be obtained through expert reviews, bench research, stakeholder or subject matter interviews, or other non-user based approaches and may not require the more time consuming approaches that require feedback from actual users.

Organizational Structure

The organizational structure established to conduct a user centered design and complementary agile software development process can have significant impact on the quality of these processes. In some organizations, the same team is given the responsibility to do both processes. This has some significant drawbacks. First, if the same team is to do both the user centered design process (come up with a design) as well as the agile software development process (implement the design) there is an inherent conflict of interest with in the team. Since the team is responsible for building the product, they are unlikely to design a product that is difficult to build so compromises will again be made in favor of a simpler software solution. In addition, the skill set associated with these two processes is

fundamentally different. User centered design process requires skills in human factors engineering (i.e. psychology), interaction or experience design, visual design, graphic design, content writing, and possibly industrial design. The skill set associated with an agile software development process includes software engineering, Web development, system security, information security, networking, and databases. There are few if any people sufficiently skilled to be considered capable of performing both functions adequately[4]. Even if they were, they would need to be a member of one team or the other for the reasons just identified (to avoid the conflict of interest).

In some organizations, the user centered design process is a staffed by a separate group with the appropriate skills but one that reports directly to the software development team. This creates a different conflict of interest in the management of these two teams. As it was eloquently put, “there is a reason we don’t ask one of the lawyers to service the judge”. If the user centered design team reports to the software development team, decisions that affect both parties will once again tend to favor the software development process since that is the primary focus of the team lead.[5]

A more appropriate arrangement between the two processes is to create separate teams, each with a responsibility for one of the two processes, that both report to a neutral authority. In larger organizations, this neutral authority is a systems engineer. The system engineer is neither a member of the user centered design team nor a member of the software development team. System engineers are members of the management team responsible for overall project completion (total project schedule and cost for example). Their role as project managers overseeing both teams would be to strike an appropriate balance between the desire is for a more optimal interface design versus the cost and schedule impact of implementing design[6] in such an arrangement, the natural tension between these two teams is both expected and desirable since it clearly identifies where trade-offs between the two teams occurs.

---

[4] It is also unlikely that any single team member would be sufficiently skilled in all of the areas within either of these processes let alone both of them, though in smaller organizations a single person may be expected to be reasonably capable in multiple elements of their selected process.

[5] It is possible that does not seem to be a very common occurrence that these rules could be reversed–that the software development team would report to the user centered design team. In such a case the reverse conflict of interest would also exist and is not a recommended approach.

[6] In very large organizations and larger projects, the System Engineer may be responsible for elements outside of the software development process such as manpower, personnel, training, facilities, etc.

About the Author

Bill Killam is a Human Factors Engineer with over 30 years of experience. He holds professional certification in Human Factors Engineering from the Board of Certification in Professional Ergonomics and is an active member of the American Psychological Association, the Human Factors and Ergonomic Society, the Usability Professionals Association, and the Association of Computing Machinery. He is the president and principle consultant at User-Centered Design, Inc., a Human Factors Engineering consultancy in Northern Virginia. He is also a professor in the College of Systems Engineering and Operations Research at George Mason University, the College of Information Studies at the University of Maryland, and the School of Library and Information Science at Catholic University of America.